Requirements Elicitation

Introduction into Software Engineering
Lecture 4
25. April 2007

Bernd Bruegge
Applied Software Engineering
Technische Universitaet Muenchen

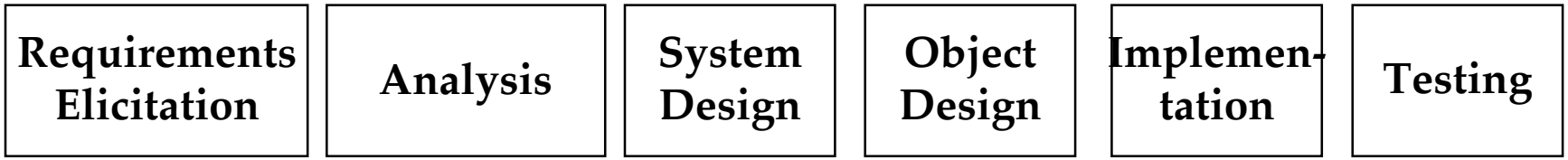
Outline

- Motivation: Software Lifecycle
- Requirements elicitation challenges
- Problem statement
- Requirements specification
  - Types of requirements
- Validating requirements
- Summary

Software Lifecycle Definition

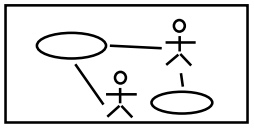
- Software lifecycle
  - Models for the development of software
  - **Functional model:**
    - Set of activities and their dependency relationships to each other to support the development of a software system
    - Examples:
      - Analysis, Design, Implementation
- Typical Lifecycle questions:
  - Which activities should I select when I develop software?
  - What are the dependencies between activities?
  - How should I schedule the activities?

Software Lifecycle Activities



Software Lifecycle Activities...and their models

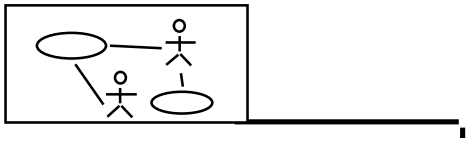
Requirements Elicitation	Analysis	System Design	Object Design	Implementation	Testing



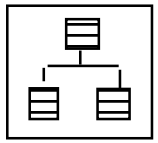
Use Case Model

Software Lifecycle Activities...and their models

Requirements Elicitation	Analysis	System Design	Object Design	Implementation	Testing



Expressed in terms of

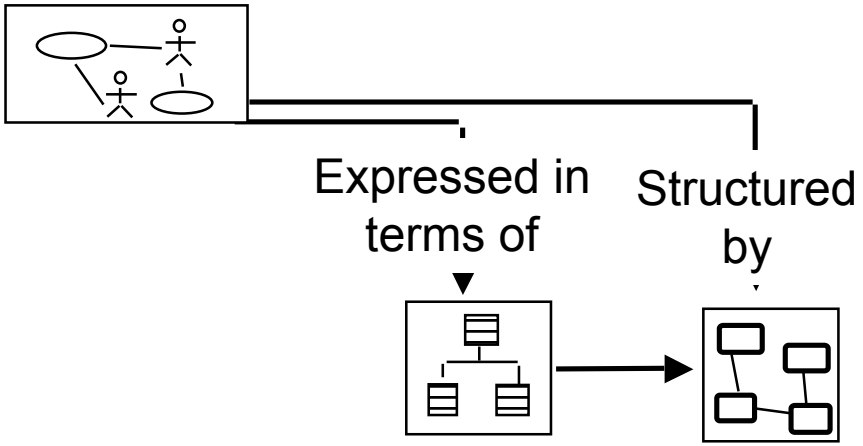


Use Case Model

Application Domain Objects

Software Lifecycle Activities...and their models

Requirements Elicitation	Analysis	System Design	Object Design	Implementation	Testing



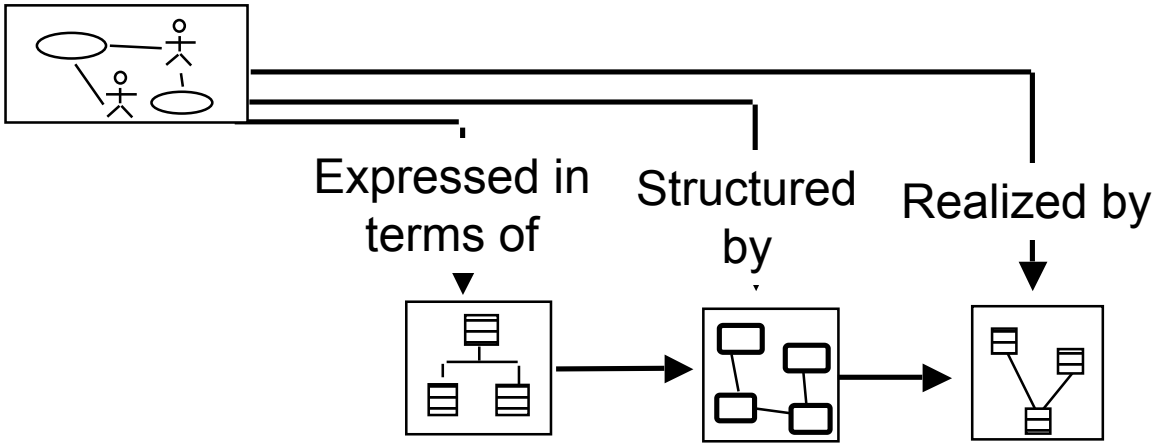
Use Case Model

Application Domain Objects

Sub-systems

Software Lifecycle Activities...and their models

Requirements Elicitation	Analysis	System Design	Object Design	Implementation	Testing



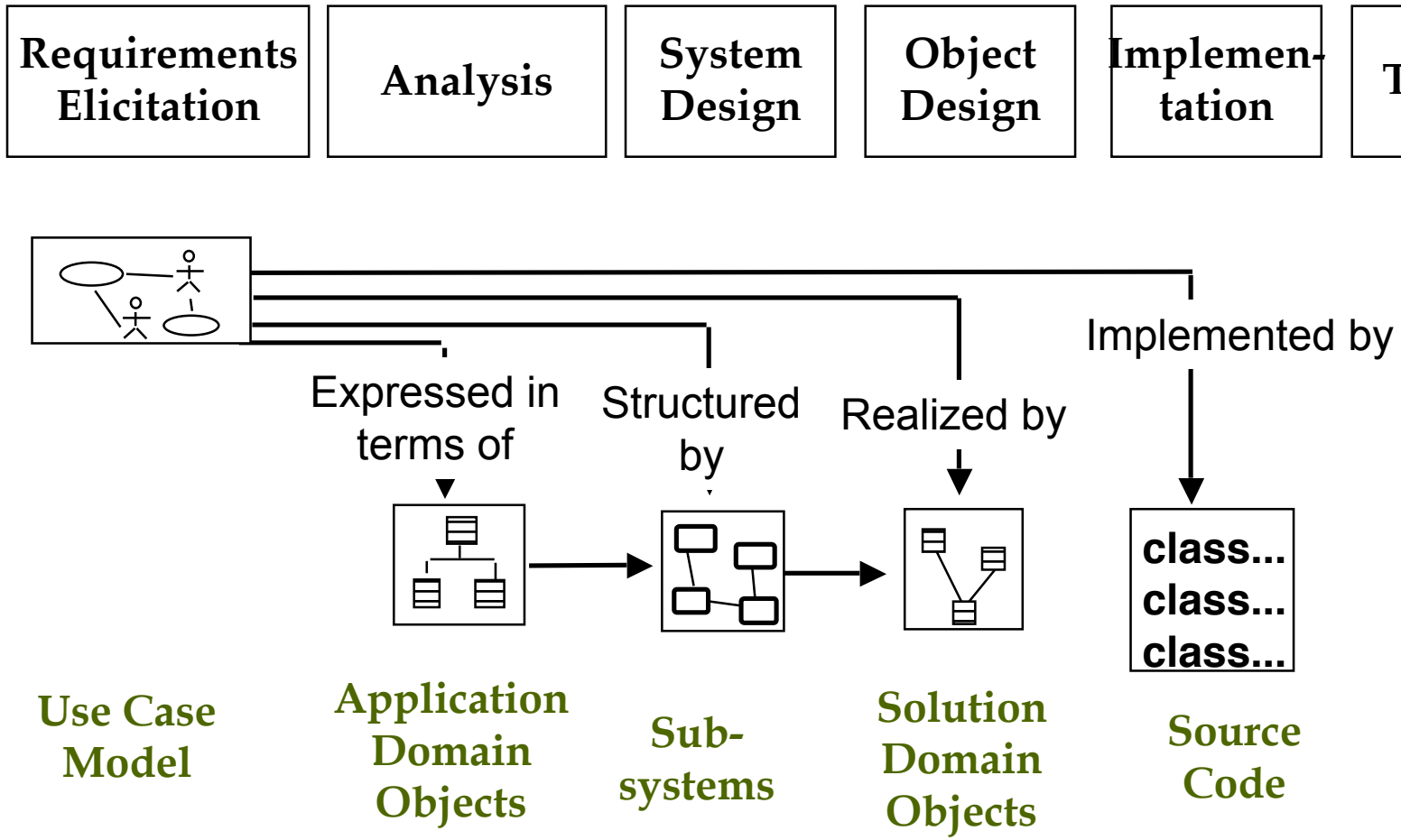
Use Case Model

Application Domain Objects

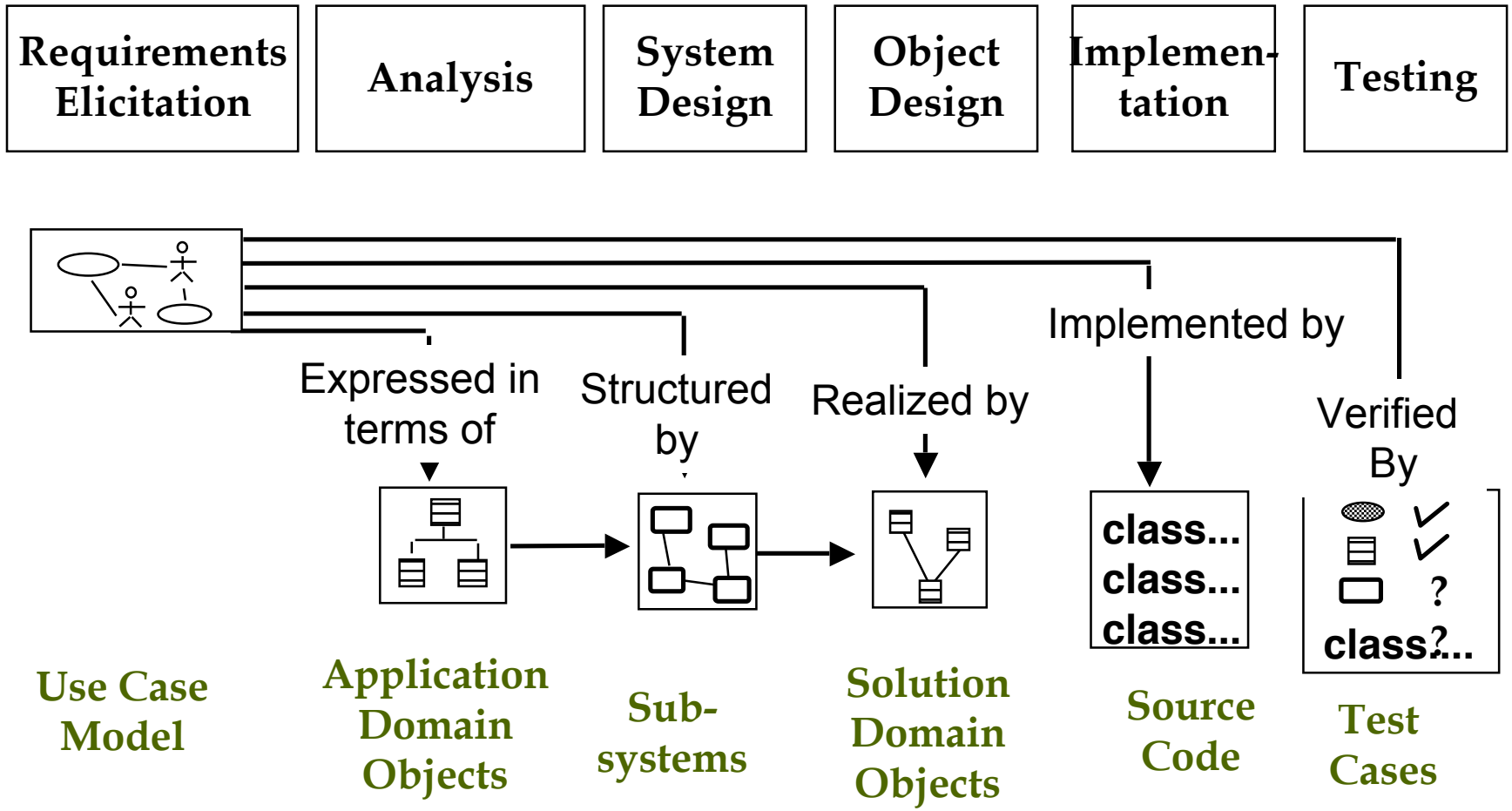
Sub-systems

Solution Domain Objects

Software Lifecycle Activities...and their models



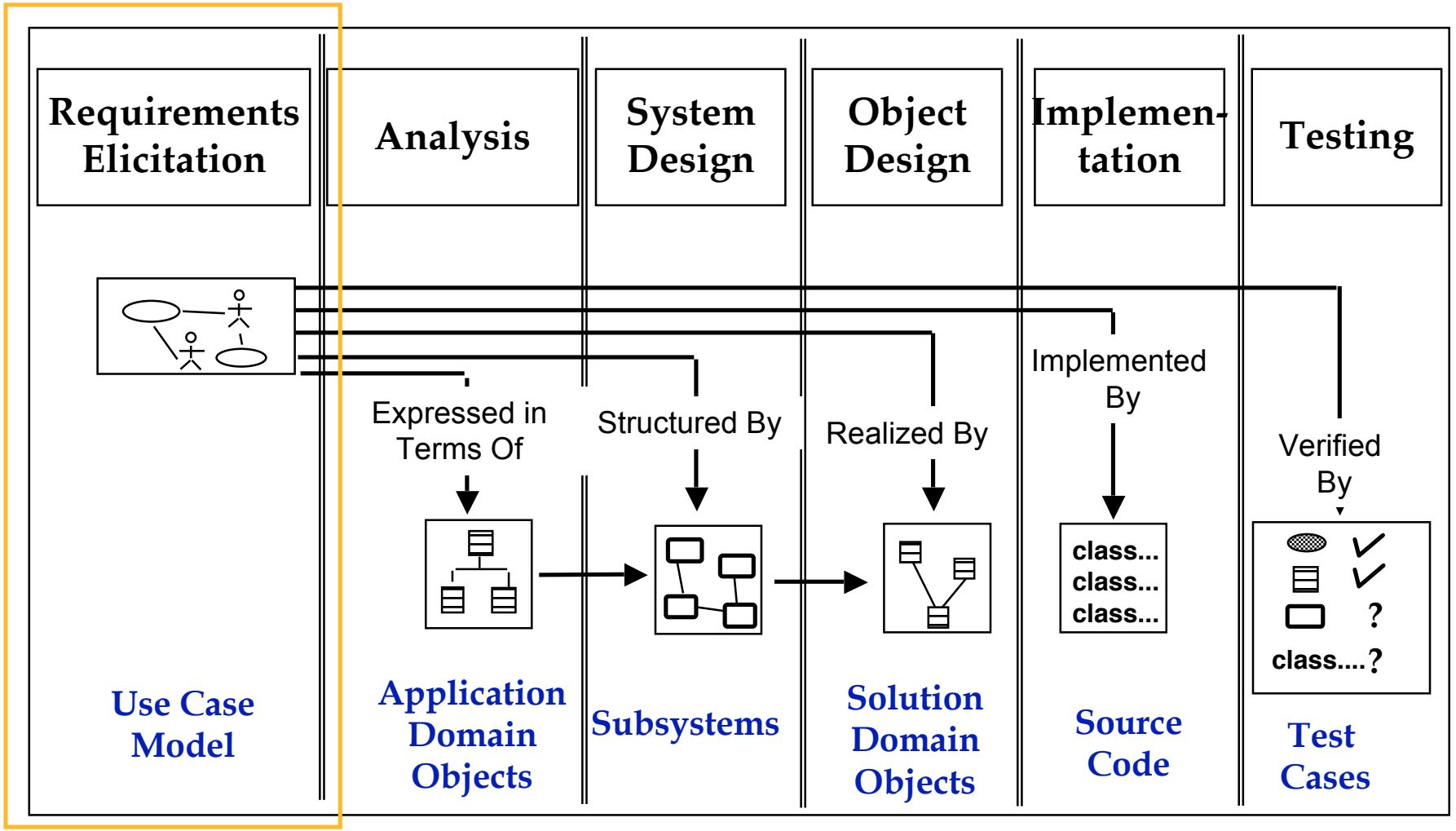
Software Lifecycle Activities...and their models



What is the best Software Lifecycle?

- Answering this question is the topics of the lecture on software lifecycle modeling
- For now we assume we have a set of predefined activities which I present to you in a linear way:
  - Today we focus on the activity requirements elicitation

Software Lifecycle Activities



What does the Customer say?

First step in identifying the Requirements: System identification

- Two questions need to be answered:
  1. How can we identify the purpose of a system?
  2. What is inside, what is outside the system?
- These two questions are answered during requirements elicitation and analysis
- Requirements elicitation:
  - Definition of the system in terms understood by the customer ("Requirements specification")
- Analysis:
  - Definition of the system in terms understood by the developer (Technical specification, "Analysis model")
- Requirements Process: Contains the activities Requirements Elicitation and Analysis.

Techniques to elicit Requirements

- Bridging the gap between end user and developer:
  - **Questionnaires:** Asking the end user a list of pre-selected questions
  - **Task Analysis:** Observing end users in their operational environment
  - **Scenarios:** Describe the use of the system as a series of interactions between a concrete end user and the system
  - **Use cases:** Abstractions that describe a class of scenarios.

Requirements Elicitation: Difficulties and Challenges

- Communicating accurately about the domain and the system
  - People with different backgrounds must collaborate to bridge the gap between end users and developers
    - Client and end users have application domain knowledge
    - Developers have solution domain knowledge
- Identifying an appropriate system (Defining the system boundary)
- Providing an unambiguous specification
- Leaving out unintended features

=> 3 Examples.

Defining the System Boundary is difficult

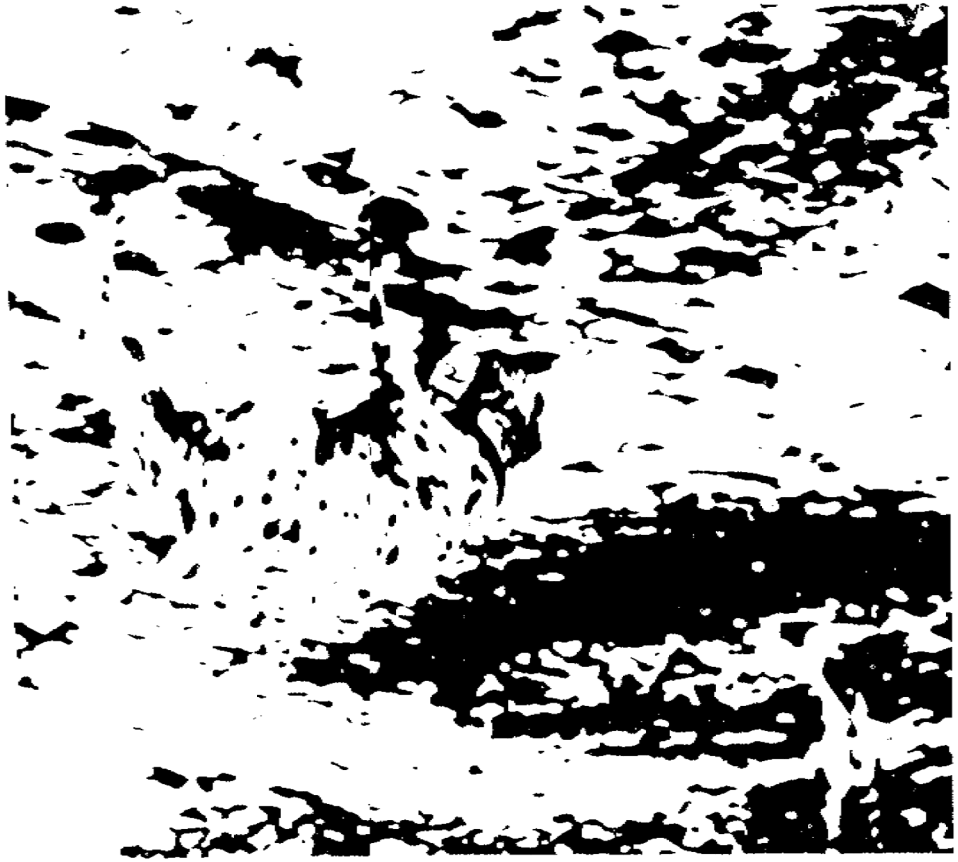
What do you see here?

Defining the System Boundary is difficult

What do you see now?

Defining the System Boundary is difficult

What do you see now?



Example of an Ambiguous Specification

During a laser experiment, a laser beam was directed from earth to a mirror on the Space Shuttle Discovery

The laser beam was supposed to be reflected back towards a mountain top 10,023 feet high

The operator entered the elevation as "10023"

The light beam never hit the mountain top
What was the problem?

The computer interpreted the number in miles...

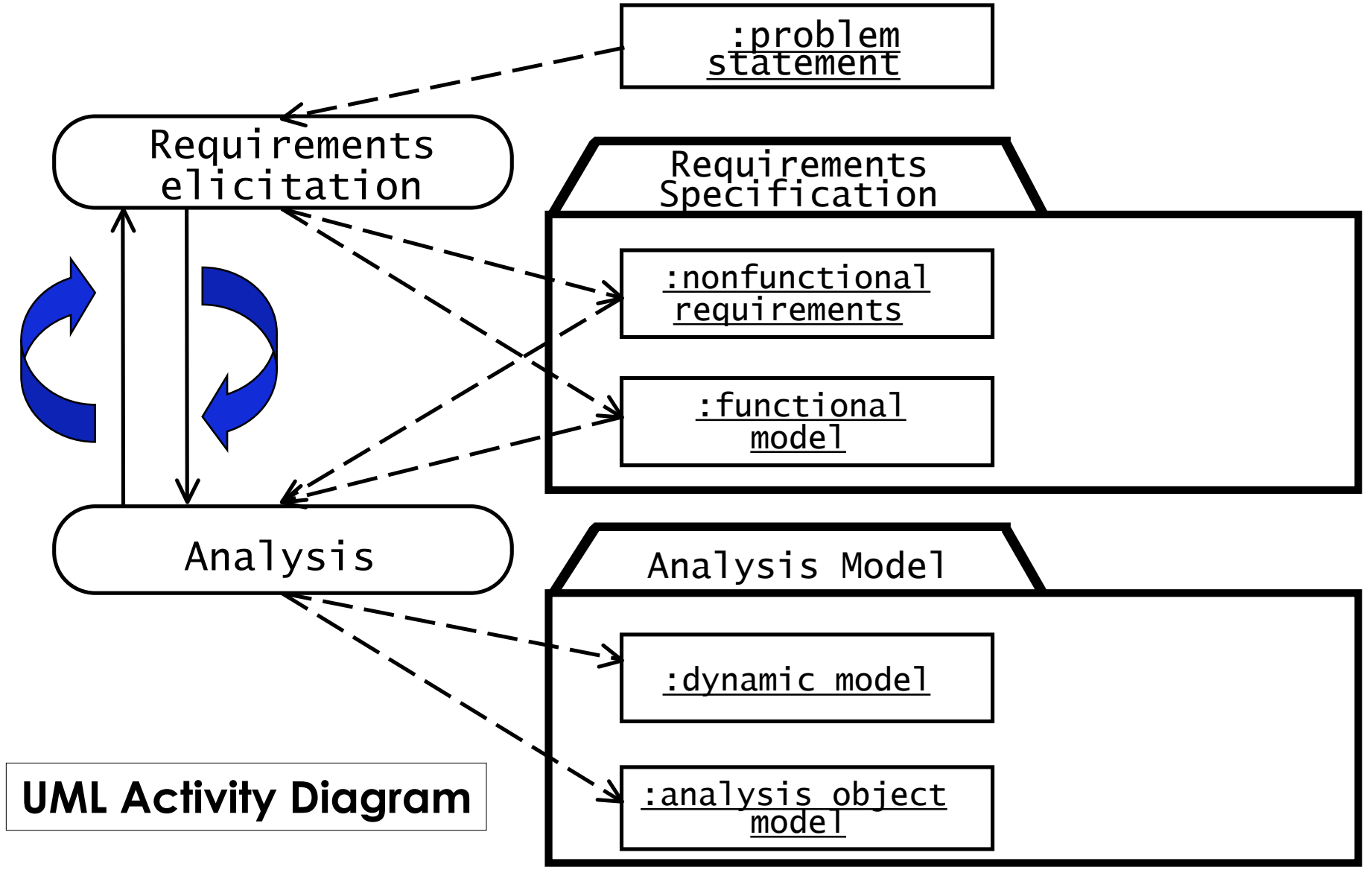
Example of an Unintended Feature

From the News: London underground train leaves station without driver!

What happened?

- A passenger door was stuck and did not close
- The driver left his train to close the passenger door
  - He left the driver door open
  - He relied on the specification that said the train does not move if at least one door is open
- When he shut the passenger door, the train left the station without him
  - The driver door was not treated as a door in the source code!

Requirements Process



UML Activity Diagram

Requirements Specification vs Analysis Model

Both focus on the requirements from the user's view of the system

- The requirements specification uses natural language (derived from the problem statement)
- The analysis model uses a formal or semi-formal notation
  - We use UML.

Types of Requirements

- Functional requirements
  - Describe the interactions between the system and its environment independent from the implementation

    "An operator must be able to define a new game. "
- Nonfunctional requirements
  - Aspects not directly related to functional behavior.

    "The response time must be less than 1 second"
- Constraints
  - Imposed by the client or the environment
    - "The implementation language must be Java "
  - Called "Pseudo requirements" in the text book.

Functional vs. Nonfunctional Requirements

Functional Requirements

- Describe user tasks that the system needs to support
- Phrased as actions
  - "Advertise a new league"
  - "Schedule tournament"
  - "Notify an interest group"

Nonfunctional Requirements

- Describe properties of the system or the domain
- Phrased as constraints or negative assertions
  - "All user inputs should be acknowledged within 1 second"
  - "A system crash should not result in data loss".

Types of Nonfunctional Requirements

Quality requirements

Constraints or
Pseudo requirements

Types of Nonfunctional Requirements

- Usability
- Reliability
  - Robustness
  - Safety
- Performance
  - Response time
  - Scalability
  - Throughput
  - Availability
- Supportability
  - Adaptability
  - Maintainability

Quality requirements

Constraints or
Pseudo requirements

Types of Nonfunctional Requirements

- Usability
- Reliability
  - Robustness
  - Safety
- Performance
  - Response time
  - Scalability
  - Throughput
  - Availability
- Supportability
  - Adaptability
  - Maintainability

Quality requirements

- Implementation
- Interface
- Operation
- Packaging
- Legal
  - Licensing
  - Certification
  - Regulation

Constraints or
Pseudo requirements

Some Quality Requirements Definitions

- Usability
  - The ease with which actors can use the system to perform a function
  - **Must be measurable, otherwise it is not usability but marketing ("The system is easy to use")**
    - Example: Number of steps to purchase with a web browser
- Robustness: The ability of the software system to maintain a function even if the user enters a wrong input, or if there are changes in the environment
  - Example: The system can tolerate temperatures up to 90 C
- Maintainability: The ease with which a function can be changed in accordance with the requirements
- Availability: The ratio of the expected uptime of a system to the aggregate of the expected up and down time
  - Example: The system is down not more than 5 minutes per week.

Nonfunctional Requirements: ARENA examples

- "Spectators must be able to watch a match without prior registration and without prior knowledge of the match."
  - *Usability Requirement*
- "The system must support 10 parallel tournaments"
  - *Performance Requirement*
- "The operator must be able to add new games without modifications to the existing system."
  - *Supportability Requirement*

What should not be in the Requirements?

- System structure, implementation technology
- Development methodology
  - Parnas, How to fake the software development process
- Development environment
- Implementation language
- Reusability

- It is desirable that none of these above are constrained by the client.

Requirements Validation

Requirements validation is a quality assurance step, usually performed after requirements elicitation or after analysis

- Correctness:
  - The requirements represent the client's view
- Completeness:
  - All possible scenarios, in which the system can be used, are described
- Consistency:
  - There are no requirements that contradict each other.

Requirements Validation (2)

- Clarity:
  - Requirements can only be interpreted in one way
- Realism:
  - Requirements can be implemented and delivered
- Traceability:
  - Each system behavior can be traced to a set of functional requirements

- Problems with requirements validation:
  - Requirements change quickly during requirements elicitation
  - Inconsistencies are easily added with each change
  - Tool support is needed!

Requirements for Requirements Management

- Tool support for managing requirements:
  - Store requirements in a shared repository
  - Provide multi-user access
  - Automatically create a system specification document
  - Allow change management
  - Provide traceability of the requirements throughout the artifacts of the system.

Tools for Requirements Management (2)

DOORS (Telelogic)

- Multi-platform requirements management tool, for teams working in the same geographical location. DOORS XT for distributed teams

RequisitePro (IBM/Rational)

- Integration with MS Word
- Project-to-project comparisons via XML baselines

RD-Link (http://www.ring-zero.com)

- Provides traceability between RequisitePro & Telelogic DOORS

Sysiphus (http://sysiphus.in.tum.de/)

- Research tool for the collaborative development of system models
- Participants can be geographically distributed.

Types of Requirements Elicitation

- Greenfield Engineering
  - Development starts from scratch, no prior system exists, requirements come from end users and clients
  - Triggered by user needs
- Re-engineering
  - Re-design and/or re-implementation of an existing system using newer technology
  - Triggered by technology enabler
- Interface Engineering
  - Provision of existing services in a new environment
  - Triggered by technology enabler or new market needs

Prioritizing requirements

- High priority
  - Addressed during analysis, design, and implementation
  - A high-priority feature must be demonstrated
- Medium priority
  - Addressed during analysis and design
  - Usually demonstrated in the second iteration
- Low priority
  - Addressed only during analysis
  - Illustrates how the system is going to be used in the future with not yet available technology

Requirements Analysis Document Template

1. Introduction
2. Current system
3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements
   - 3.3 Nonfunctional requirements
   - 3.4 Constraints ("Pseudo requirements")
   - 3.5 System models
     - 3.5.1 Scenarios
     - 3.5.2 Use case model
     - 3.5.3 Object model
       - 3.5.3.1 Data dictionary
       - 3.5.3.2 Class diagrams
     - 3.5.4 Dynamic models
     - 3.5.5 User interfae
4. Glossary

Section 3.3 Nonfunctional Requirements

3.3.1 User interface and human factors
3.3.2 Documentation
3.3.3 Hardware considerations
3.3.4 Performance characteristics
3.3.5 Error handling and extreme conditions
3.3.6 System interfacing
3.3.7 Quality issues
3.3.8 System modifications
3.3.9 Physical environment
3.3.10 Security issues
3.3.11 Resources and management issues

Nonfunctional Requirements (Questions to overcome "Writers block")

User interface and human factors

- What type of user will be using the system?
- Will more than one type of user be using the system?
- What training will be required for each type of user?
- Is it important that the system is easy to learn?
- Should users be protected from making errors?
- What input/output devices are available

Documentation

- What kind of documentation is required?
- What audience is to be addressed by each document?

Nonfunctional Requirements (2)

Hardware considerations

- What hardware is the proposed system to be used on?
- What are the characteristics of the target hardware, including memory size and auxiliary storage space?

Performance characteristics

- Are there speed, throughput, response time constraints on the system?
- Are there size or capacity constraints on the data to be processed by the system?

Error handling and extreme conditions

- How should the system respond to input errors?
- How should the system respond to extreme conditions?

Nonfunctional Requirements (3)

System interfacing

- Is input coming from systems outside the proposed system?
- Is output going to systems outside the proposed system?
- Are there restrictions on the format or medium that must be used for input or output?

Quality issues

- What are the requirements for reliability?
- Must the system trap faults?
- What is the time for restarting the system after a failure?
- Is there an acceptable downtime per 24-hour period?
- Is it important that the system be portable?

Nonfunctional Requirements (4)

System Modifications

- What parts of the system are likely to be modified?
- What sorts of modifications are expected?

Physical Environment

- Where will the target equipment operate?
- Is the target equipment in one or several locations?
- Will the environmental conditions be ordinary?

Security Issues

- Must access to data or the system be controlled?
- Is physical security an issue?

Nonfunctional Requirements (5)

Resources and Management Issues

- How often will the system be backed up?
- Who will be responsible for the back up?
- Who is responsible for system installation?
- Who will be responsible for system maintenance?